Live Interaction Server Side:

I am using socket.on, socket.run, and socket.emit.

Flask-SocketIP allows low latency bi-directional communication between client and server. It is compatible with client side SocketIO which is what we use to send player positions from our Game World.

I am using Flask-SocketIO to establish multiple socket connections with my clients. More specifically, I need to establish a connection with a client that’s logged in and receive constant information on their position. The Server must also broadcast or emit the position of every logged in client to each client so that they may render them in the Game World. When a client establishes a connection, the server will also emit the position stored in the database which would be their previous location before logging out.

**from** **flask** **import** Flask, render\_template

**from** **flask\_socketio** **import** SocketIO

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'secret!'

socketio = SocketIO(app)

**if** \_\_name\_\_ == '\_\_main\_\_':

socketio.run(app)

The web server is started calling socketio.run(), replacing app.run() in standard Flask development. The client must receive a page that loads the Socket.IO library to establish a connection.

https://github.com/miguelgrinberg/Flask-SocketIO/blob/2f1c322e687a8f7f080a081f042efdc6c8532123/flask\_socketio/\_\_init\_\_.py#L518

It will default to localhost and port 5000 if Host and Port are not specified. Next check the server.eio.async\_mode. The Engine IO is a transport protocol that enables real-time bidirectional communication between our server and clients.

We are given three asynchronous models. Eventlet is a concurrent networking library that gives us the WebSocket transports. Gevent has long-polling transport but does not have WebSocket support. It must rely on the gevent-websocket package or uWSGI to load our Flask application. Werkzeug can also be used but only for long-polling transport.

Messages are sent and received as events. Flask-SocketIO must register handlers for these events. For our sever we used events that’s just had 1 argument, a position JSON.

**@socketio**.on('my event')

**def** handle\_my\_custom\_event(json):

print('received json: ' + str(json))

https://github.com/miguelgrinberg/Flask-SocketIO/blob/2f1c322e687a8f7f080a081f042efdc6c8532123/flask\_socketio/\_\_init\_\_.py#L260

We use the emit() function to return messages backs to the client. It will emit the same namespace as the incoming message by default.

https://github.com/miguelgrinberg/Flask-SocketIO/blob/2f1c322e687a8f7f080a081f042efdc6c8532123/flask\_socketio/\_\_init\_\_.py#L403

It’s important that out website broadcast our message so that all logged in clients can see the position of everyone in real-time. To turn a regular emit() to a broadcast we set the optional broadcast argument to True. This will message all the clients connected to the socket, not just the sender.

**@socketio**.on('my event')

**def** handle\_my\_custom\_event(data):

emit('my response', data, broadcast=**True**)

Clients connected to the namespace will receive the message.

https://github.com/miguelgrinberg/Flask-SocketIO/blob/2f1c322e687a8f7f080a081f042efdc6c8532123/flask\_socketio/namespace.py#L28

License: MIT License

Permission is granted free of charge to anyone obtaining the code to copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software. The creators are also not liable for any copyright issues or problems resulting from our code and usage.

<https://github.com/miguelgrinberg/Flask-SocketIO/blob/master/flask_socketio/__init__.py>

<https://flask-socketio.readthedocs.io/en/latest/>

<https://python-engineio.readthedocs.io/en/latest/index.html>